Gromacs - Bug #1255
bond constraint contribution to dhdl not being added.
05/17/2013 08:39 PM - Michael Shirts

<table>
<thead>
<tr>
<th>Status:</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>High</td>
</tr>
<tr>
<td>Assignee:</td>
<td>Michael Shirts</td>
</tr>
<tr>
<td>Category:</td>
<td>mdrun</td>
</tr>
<tr>
<td>Target version:</td>
<td>4.6.3</td>
</tr>
<tr>
<td>Affected version - extra info:</td>
<td>Affected version: 4.6</td>
</tr>
</tbody>
</table>

Description
See subject. The dhdl of bond constraint is being added to enerd->term[F_DVDL_BONDED], but then erased before being summed.

Proposed fix is to resurrect the F_DHDL_CON term (renamed F_DVDL_CONSTR) and use it to save this information, and THEN sum it into the bond component.

A fix has been (lightly) tested and will be posted to gerrit very soon.

Associated revisions
Revision 5d9070d9 - 05/21/2013 04:55 PM - Michael Shirts
Fixed problem with bond constraint dhdl being left out
At some point, the dhdl from the holonomic bond constraints got left out, because the term got zeroed before being added to the bonded energy.
Also, I made some of the variable names involved more descriptive to clarify the logic.
Removed velocity constraint calls, as don't contribute to dhdl.
Fixes redmine #1255
Change-Id: l2eebf357d1a922b7636dc015b09f286827283dd0

Revision 714b43e2 - 05/30/2013 10:19 AM - Michael Shirts
Fixing the constraint dhdl term in shake
Addresses and corrects new issues with #1255, with additional comments explaining what else may need to be done in the future to make sure that the theory of the changes are completely correctly founded. See the redmine entry #1255 to see the numerical validation of the current choices. Also adds possible missing constraint dhdl contribution.
Fixes #1268
Change-Id: lbe3980c2e265cced49b03b966e02143c5cf9f07c
Revision 513813b5 - 10/09/2014 03:59 PM - Mark Abraham

Clean up cshake() and resolve old issues

C++ conversion in Id190e36 exposed some "interesting" use of the variable toler caused by an ancient bug that has since been resolved. Investigation showed that variable never meant anything to do with a tolerance!

Renamed a bunch of variables for clarity and consistency. Documented the core function. Clarified code comments and manual sections accordingly.

Added some unit tests while trying to understand what things were actually going on.

Refs #1255

Change-Id: i5b5eee0b8f3f4761ce9cb681dbdb0d4526a6761d

History

#1 - 05/17/2013 10:56 PM - Mark Abraham
The parade of mdrun fixes for 4.6.2 continues :-) 

#2 - 05/18/2013 12:30 AM - Michael Shirts
But everybody loves a parade!

#3 - 05/20/2013 01:54 PM - Mark Abraham
- Status changed from New to Accepted

#4 - 05/20/2013 01:54 PM - Mark Abraham
- Status changed from Accepted to In Progress

#5 - 05/20/2013 01:54 PM - Mark Abraham
- Status changed from In Progress to Fix uploaded

#6 - 05/20/2013 02:56 PM - Sziárd Páll
Is this really a "normal" priority bug? Should it not be at least high priority and perhaps announced so people know about it and can redo their simulations if necessary?

#7 - 05/20/2013 03:22 PM - Michael Shirts
- Priority changed from Normal to High

Probably should be higher. Raised.
Further testing showed that although these differences are now summed, they are not yet calculated correctly. Validation was against changing a bond length, which should have constraint free energy \( \propto k_B T \ln (\text{bond}_{\text{old}}/\text{bond}_{\text{new}}) \), and thus \( \frac{\text{dh}}{\text{dl}} = \frac{\text{dr}}{\text{dl}}/r \), where \( r = \text{bond}_{\text{old}} + \lambda(\text{bond}_{\text{new}} - \text{bond}_{\text{old}}) \).

- shake dhdl has an extra factor of the bond length in the denominator that was added at some point in the past (likely a few versions ago). I am figuring out where to put this exactly and documenting in the code so the derivations are correct (I know WHAT the error is, but not where it was made yet). Lincs dhdl has the correct dependence on \( r \).

- Both shake and lincs differ by a factor of 2 between when applied to md-vv and md (not clear which is correct yet, appears to be md-vv, but I need to double check my derivation). I will figure out which one is correct and put the documentation of this in comments in the code. Free energy contribution, SHOULD be (I think) \( 2^* k_B T \ln (\text{bond}_{\text{old}}/\text{bond}_{\text{new}}) \), while md gives \( 4^* k_B T \ln (\text{bond}_{\text{old}}/\text{bond}_{\text{new}}) \), but I need to double check this.

I hope to finish this today, but requires being extra careful, so may not be until over the weekend.

I've fixed the terms now, as validated by a thermodynamic cycle below -- this allowed me to address the ambiguity in the constant term in front of the correction.

Difference in solvation of a 0.5 and 0.2 nm dipole carried out two ways:

1. solvation difference between long and short
2. difference in change in free energy in solution and in vacuum:

   solv-long - solv-short = (96.397 + 0.163) - (56.424 + 0.196) = 39.94 +/- 0.25 kJ/mol
   change-solv - change-vac = (43.597 + 0.141) - (3.321 + 0.005) = 40.27 +/- 0.14 kJ/mol

One issue: the md-vv constraint is smaller by a factor of 1/2, because only 1/2 the velocity term is uncorrected. The fastest fix is to simply double the constraint contribution. In the limit of small timesteps, this is completely correct, but it is unclear if there is noticeable error over longer timesteps. A first pass over 10 ns show that there probably isn't any noticeable error in this. The other way is to go back and compute the dhdl due to the forces in the velocity step, but this would require a lot more deeper work. This approach is, at the worst, correct to the limit of statistical error, and I'd rather make sure this gets in 4.6.2, and make fine adjustments later than not have it in at all.
Just to clarify about whether the approximation for md-vv (doubling the position restraint correction term) is correct, here’s the test with changing the dipole length from 0.5 to 0.2 nm with md, md-vv, and sd, 10 ns simulation. Note there is one sd simulation (lambda state = 1) that is running really slowly for some pinning issue, affecting the overall uncertainty-- I will fix it when it finishes. I don’t think it affects the conclusions.

<table>
<thead>
<tr>
<th></th>
<th>md:</th>
<th>md-vv:</th>
<th>sd:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 — 1</td>
<td>1.760 ± 0.018</td>
<td>1.752 ± 0.018</td>
<td>1.680 ± 0.078</td>
</tr>
<tr>
<td>1 — 2</td>
<td>1.831 ± 0.018</td>
<td>1.805 ± 0.018</td>
<td>1.764 ± 0.067</td>
</tr>
<tr>
<td>2 — 3</td>
<td>2.307 ± 0.019</td>
<td>2.267 ± 0.019</td>
<td>2.315 ± 0.024</td>
</tr>
<tr>
<td>3 — 4</td>
<td>2.887 ± 0.019</td>
<td>2.831 ± 0.019</td>
<td>2.887 ± 0.022</td>
</tr>
<tr>
<td>4 — 5</td>
<td>3.412 ± 0.019</td>
<td>3.386 ± 0.019</td>
<td>3.415 ± 0.021</td>
</tr>
<tr>
<td>5 — 6</td>
<td>4.225 ± 0.018</td>
<td>4.235 ± 0.018</td>
<td>4.226 ± 0.020</td>
</tr>
<tr>
<td>6 — 7</td>
<td>5.246 ± 0.017</td>
<td>5.239 ± 0.018</td>
<td>5.288 ± 0.020</td>
</tr>
<tr>
<td>7 — 8</td>
<td>6.361 ± 0.017</td>
<td>6.341 ± 0.017</td>
<td>6.335 ± 0.020</td>
</tr>
<tr>
<td>8 — 9</td>
<td>7.340 ± 0.016</td>
<td>7.288 ± 0.016</td>
<td>7.287 ± 0.019</td>
</tr>
<tr>
<td>9 — 10</td>
<td>8.085 ± 0.016</td>
<td>8.032 ± 0.019</td>
<td>8.044 ± 0.018</td>
</tr>
</tbody>
</table>

---

TOTAL: | 43.455 ± 0.069 | 43.178 ± 0.070 | 43.221 ± 0.154 |

So, they appear to be almost statistically indistinguishable. The approximation for md-vv (if it is an approximation) gives results that appear to be well within noise for sd, and close to noise to md-vv (2.6 sigma). This should be good enough for essentially all situations.

#12 - 05/27/2013 05:10 AM - Mark Abraham
Tabs are evil. Fixed :-)

#13 - 05/31/2013 10:46 AM - Mark Abraham
- Target version changed from 4.6.2 to 4.6.3

#14 - 06/26/2013 12:35 AM - Mark Abraham
- Status changed from Fix uploaded to Resolved

#15 - 12/16/2013 05:41 PM - Rossen Apostolov
- Status changed from Resolved to Closed

#16 - 08/16/2014 09:58 AM - Mark Abraham
While converting files to C++, I unearthed the cause of the bug that led to this fix.

Michael inserted some braces wrongly in http://redmine.gromacs.org/projects/gromacs/repository/revisions/a6ee084bcf3ac87d9dd493702b1c773d86506bdc/diff/src/mdlib/shakef.c that broke some FEP functionality at the end of vec_shake, which led to him later re-implementing that in bshakef (see...
Berk's original implementation re-used the toler variable to post-multiply the Lagrangian multipliers by the constraint distance (weighted by FE-lambda in that case), but those multipliers are unused in the non-FE case. I don't think there is any use for that other than computing dH/dl, but we should clear that up rather than returning a Lagrangian that is scaled by the constraint distance. Michael's subsequent changes have converged on an implementation equivalent to Berk's original one, but with clarity in what value lagr[] actually holds.

Michael also observed that there is a factor of two that seems to be missing in the SHAKE case. The original Ryckaert SHAKE paper derives the constraints based on $(r_i - r_j)^2 - d^2 = 0$, but the LINCS paper uses $|r_i - r_j| - d = 0$. GROMACS documentation is inconsistent, with neither manual section 3.6.1 and 3.6.2 directly agreeing with section 4.5. The implementation seems to use the LINCS definition in the SHAKE case for computing dH/dl, which offhand seems OK so long as it is consistently implemented (and documented). So, is it? And how should we fix the docs?

Clariﬁed - the cshake/crattle routines return the Lagrangian multipliers divided by their respective constraint distances, so the post-multiplication was simply ﬁnishing the job of computing the Lagrangian multipliers.

Michael also observed that there is a factor of two that seems to be missing in the SHAKE case. The original Ryckaert SHAKE paper derives the constraints based on $(r_i - r_j)^2 - d^2 = 0$, but the LINCS paper uses $|r_i - r_j| - d = 0$. GROMACS documentation is inconsistent, with neither manual section 3.6.1 and 3.6.2 directly agreeing with section 4.5. The implementation seems to use the LINCS definition in the SHAKE case for computing dH/dl, which offhand seems OK so long as it is consistently implemented (and documented). So, is it? And how should we fix the docs?

Fixed docs, and made derivation in 4.5 make more sense, which clarifies why the code was correct all along. Note the distinction in the case of SHAKE between the quadratic constraint equation, and the linear lambda-interpolation.